

CS 386/586 Winter 2012  
Assignment 3  
Possible answers

1. Write an SQL query that lists the agents (id, first, last) who have been on a 'Secret' or 'Top Secret' mission that succeeded.

Write this query three ways, once using EXISTS, once using IN, and once **without** using any of the special predicates in the WHERE clause that deal with subqueries (such as IN, NOT IN, EXISTS, etc.).

Here's a query answer that simply uses joins.

```
select a.agent_id, a.first, a.last
from  agent a join teamrel tr on a.agent_id = tr.agent_id
      join mission m on tr.team_id = m.team_id
      join securityclearance sc on m.access_id = sc.sc_id
where m.mission_status = 'success' and (sc.sc_level = 'Secret' or sc.sc_level = 'Top Secret')
```

Here's a query that uses IN. We just need to find an agent whose agent id appears in the information that we gather from a subquery. *Note agent does not need to be joined in in the nested query. My query is not incorrect – but it has some unnecessary work in it. Note that some query optimizers are able to eliminate “unused” tables.*

```
select a.agent_id, a.first, a.last
from  agent a
where a.agent_id in
      (select a.agent_id
       from  agent a join teamrel tr on a.agent_id = tr.agent_id
           join mission m on tr.team_id = m.team_id
           join securityclearance sc on m.access_id = sc.sc_id
       where m.mission_status = 'success' and (sc.sc_level = 'Secret' or
                                              sc.sc_level = 'Top Secret'))
```

Here's a query that uses EXISTS. The blue query is similar to the red one above – except for the part indicated in purple. Note that this one is a correlated subquery. Here again – the inner query doesn't need to join with agent.

```
select a.agent_id, a.first, a.last
from  agent a
where EXISTS
      (select a2.agent_id
       from  agent a2 join teamrel tr on a2.agent_id = tr.agent_id
           join mission m on tr.team_id = m.team_id
           join securityclearance sc on m.access_id = sc.sc_id
       where m.mission_status = 'success' and (sc.sc_level = 'Secret' or
```

```
sc.sc_level = 'Top Secret')
and a.agent_id = a2.agent_id)
```

Describe that query or queries that you issued or other steps that you performed to convince yourself that all three queries returned the exact same answer.

You could check to see which teams had a security clearance of Top Secret or Secret. Then check to see which of those had a mission\_status of success. Then confirm that you were picking put agents who were on those teams.

Write a relational algebra query that is equivalent to any of these three SQL queries.

It's probably easiest to write the join query in relational algebra. You can even write using cross product followed by select like this

```
 $\pi$  a.agent_id, a.first, a.last ( $\sigma$  a.agent_id = tr.agent_id and tr.team_id = m.team_id and m.access_id = sc.sc_id and m.mission_status = 'success' and (sc.sc_level = 'Secret' or sc.sc_level = 'Top Secret') (agent a X teamrel tr X mission X securityclearance sc))
```

2. Write an SQL query that lists the country and city for which the agents from that country/city have the lowest average salary (compared to the average salary for agents from other cities and countries), along with the average salary.

```
select country, city, avg(salary)
from agent
group by country, city
having avg(salary) = (select min(avsal)
                    from (select avg(salary) as avsal
                          from agent
                          group by country, city) as y)
```

3. Write a query that counts the number of agents that are not on a team. Your query answer should be just a count (a single value).

```
Select count(*)
From (select distinct agent_id from agent except select distinct agent_id from teamrel) as x)
```

count

368

Describe that query or queries that you issued or other steps that you performed to convince yourself that your returned the correct answer.

```
Select count(*) from agent
```

There are 662 agents.

```
select count(distinct agent_id) from teamrel
```

There are 294 agent id appearing the teamrel. So,  $662 - 294 = 378$

4. Write an SQL query that lists all of the agent\_id who speak the maximum number of languages compared to the number of languages spoken by all of the agents in the Introdb\_spy database. Note you must use your query to figure out the maximum number of languages spoken by all of the agents; you are NOT allowed to figure that out by hand and then use a constant in your query.)

This is similar to query 2. Note: we only need to use languagerel.

```
select agent_id
from languagerel
group by agent_id
having count(*) = (select max(ct)
                  from (select count(*) as ct
                        from languagerel
                        group by agent_id) as x)
```

Check it out:

```
select * from languagerel where agent_id = 106
```

lang_id	agent_id
2	106
3	106
8	106
9	106
13	106

5 row(s)

Now find the number of languages spoken by agents.  
I did an order by – just to make it easier to scan the result.

```
select agent_id, count(*)
from languagerel
group by agent_id
order by count(*), agent_id
```

If we scan to the bottom, we see that 5 is the maximum number of languages spoken by any agent and we see agent 106 with 5 languages.

5. First, explain what the following relational algebra query computes in English. Turn in your English description.

Then, write an SQL query that is equivalent to the following relational algebra query. (Hint: remember to use DISTINCT if your query answer has duplicate rows in SQL.)

$$\pi_{\text{agent\_id}} (\text{Agent} - (\sigma_{\text{city} = \text{'Paris'}} \text{Agent}))$$

This query finds the agent\_id for agents who do not live in Paris.

Try this in SQL:

```
select agent_id from agent where city != 'Paris'
```

608 row(s)

Try this:

```
select agent_id from agent
except
select agent_id from agent where city = 'Paris'
```

608 row(s)

In this case, every agent who does not live in Paris – will have a value for city that is not equal to 'Paris' so we didn't need to do an except. But an except query works.

Note we don't have to use distinct because agent\_id is a key for agent so there aren't any duplicate values in any of these queries or subqueries.

6. First, explain what the following relational algebra query computes in English. Turn in your English description.

Then, write an SQL query that is equivalent to the following relational algebra query.

$$\pi_{\text{agent\_id}} (\text{Agent} - (\sigma_{\text{city} = \text{'Paris'}} \text{Agent})) \cap \pi_{\text{agent\_id}} (\text{Agent} - \sigma_{\text{middle is NULL}} \text{Agent})$$

This query has a typo. (the last phrase should not have a period before Agent.) Also, somehow, the subscript notation got lost in my Word document. I've put the subscript formatting back into this query.

The first main query finds agents who do not live in Paris. It is then intersected with agents who actually have a middle name. So, the query finds agents who do not live in Paris but actually have a middle name that is not NULL.

A direct translation of the relational algebra to SQL results in this:

```
(select agent_id from agent
except
select agent_id from agent where city = 'Paris')
intersect
(select agent_id from agent
except
select agent_id from agent
where middle is null)
```

198 row(s)

One of those agents is agent id 866.  
If use this query:  
select \* from agent where agent\_id = 866

I get this answer:

agent_id	first	middle	last	address	city	country	salary	clearance_id
866	George	W	Kennedy	20 74th Avenue	Tokyo	Japan	58094	3

1 row(s)

And we see that he does NOT live in Paris and he DOES have a value for middle. You could check a couple of other agents as well. You could also find an agent that has a NULL value of middle and check to make sure that he or she does NOT appear in the answer.

7. In relational algebra, suppose I have a table completed (s-id, c-id) which indicates that a student (identified by s-id) has completed a course (identified by c-id). Suppose I have another table Required-course (c-id) which lists the ids of the required courses. If I want to find all student ids that have completed all courses in the Required-course table, I can write in in relational algebra this way:

Completed  $\div$  Required-course (or Completed / Required-course)

I can also write it this way – without using the divide operator in relational algebra like this:

$\pi_{s-id}(\text{Completed}) - \pi_{s-id}((\pi_{s-id} \text{Completed}) \times \text{Required-course}) - \text{Completed}$

Using this as a guide to how to compute a divide, write an SQL query for the `introdb_spy` database that finds the `agent_id` for agents that speak all languages in the `Language` table. Note: there may not be any agents that speak all of those languages but you can write and execute a query that finds agents who do.

```
select a.agent_id
from agent a
where NOT EXISTS
  ((select lang_id from language)
  except
  (select lang_id from languagerel lr where lr.agent_id = a.agent_id))
```

No rows found.

8. Modify your SQL query from the previous question to find all agents that speak all of these languages: French, German, and English. Hint: for this query, you might need to use a subquery or a table-valued expression instead of the `language` table.

Turn in your SQL query, (at most) the first five rows of your query answer, and the total number of rows in your query answer.

```
select a.agent_id from agent a
where NOT EXISTS
  ((values ('French'), ('German'), ('English'))
  except
  select language
  from languagerel lr join language l on lr.lang_id = l.lang_id
  where lr.agent_id = a.agent_id)
```

agent_id
95

1 row(s)

Try this:

```
select agent_id, language
from languagerel lr join language l on lr.lang_id = l.lang_id
order by agent_id
```

Scan down to see what agent 95 speaks.

95	Portuguese
95	English
95	German
95	Russian
95	French

We see that agent 95 speaks all three of the desired languages.

Try this:

```
select agent_id, language
from languagerel lr join language l on lr.lang_id = l.lang_id
where language = 'French' or language = 'German' or language = 'English'
order by agent_id
```

If you scan down the list you see that no agent number appears three time in this list except for agent 95. Here's a sample.

189	German
191	German
193	French
199	German
202	German
205	German
207	German
209	German
210	German
210	French
212	French
219	French
227	French
228	German
232	French
235	German
236	German
236	French

9. Write an SQL query that finds the teams where at least one of the agents speaks German.

This one is easy; all you need to do is join teamrel to languagerel to language and restrict the language to German.

```
select distinct team_id
from teamrel join languagerel using (agent_id) join language using(lang_id)
where language = 'German'
```

A join like this will include the team\_id as long as it appears at least once with the right tuples in the join that satisfy the WHERE clause. It makes sense to use DISTINCT for this query so that we list teams just once.

34 row(s)

10. Write an SQL query that finds the teams where none of the agents speak German.

This one requires a NOT EXISTS. We need to find all of the agents on a team and then make sure that none of the agents speak German.

Notice that we also need to write a correlated query so that we do the above for each team in the database.

```
select distinct t.team_id
from team t
where
NOT EXISTS
(select tr.agent_id from teamrel tr join languagerel lr on tr.agent_id = lr.agent_id join language l
on lr.lang_id = l.lang_id
where tr.team_id = t.team_id and l.language = 'German')
```

team_id
3
5
8
11
15
18
32
42

8 row(s)

Note: we can find the agents on team 15 this way:

```
select agent_id from teamrel where team_id = 15
```

```
agent_id
```

```
338
```

```
352
```

```
353
```

```
408
```

```
458
```

```
741
```

```
754
```

```
755
```

```
860
```

```
1097
```

Then find all the languages that these agents speak.

```
select tr.agent_id, l.language from teamrel tr join languagerel lr on tr.agent_id = lr.agent_id
```

```
join language l on lr.lang_id = l.lang_id
```

```
where tr.team_id = 15
```

```
order by language, agent_id
```

We scan the list and see that no one speaks German.

agent_id	language
741	Arabic
754	Arabic
755	Arabic
352	Bengali
408	Bengali
1097	Bengali
353	Cherokee
741	Chinese
352	Farsi
860	Farsi
1097	Farsi
860	French
408	Hebrew
755	Hebrew
1097	Hebrew
458	Hindi
1097	Hindi
458	Japanese
353	Korean
352	Malay
754	Malay

860	Malay
353	Polish
352	Portuguese
353	Portuguese
408	Portuguese
741	Portuguese
754	Portuguese
338	Russian
408	Russian
741	Russian
755	Spanish
353	Turkish
1097	Vietnamese

34 row(s)

Bonus: Write an equivalent query in relational algebra.

$(\pi_{\text{team\_id}} \text{Team}) - \pi_{\text{team\_id}} (\sigma_{l.\text{language} = \text{'German'}}$

$(\text{language} l) \bowtie_{l.r.\text{agent\_id} = \text{tr}.\text{agent\_id}} \text{teamrel } \text{tr} \bowtie_{l.r.\text{lang\_id} = l.\text{language\_id}}$